TP 2 Data Mining: Naive Bayes Classifier with Numpy (OPTIONAL - NOT GRADED)

This TP is optional. In this TP, you will implement the Naive Bayes classifier algorithm with the help of **ChatGPT**. The Naive Bayes classifier is a simple but effective algorithm for classification that is widely used in machine learning and data science. It is based on the Bayes' theorem and the assumption of feature independence, which makes it particularly suitable for high-dimensional and sparse datasets.

Theory

Naive Bayes is a classification algorithm that uses Bayes' theorem to predict the probability of a particular class label given some input features. Bayes' theorem states that:

$$P(y|x_1, x_2, \ldots, x_n) = \frac{P(y)P(x_1, x_2, \ldots, x_n|y)}{P(x_1, x_2, \ldots, x_n)} \tag{1}$$

where $P(y)$ is the prior probability of class y, $P(x_1, x_2, \ldots, x_n|y)$ is the likelihood of the input features given the class label y, and $P(x_1, x_2, \ldots, x_n)$ is the marginal likelihood of the input features.

The "naive" assumption in Naive Bayes is that the input features are conditionally independent given the class label. That is:

$$P(x_1, x_2, \ldots, x_n|y) = \prod_{i=1}^{n} P(x_i|y) \tag{2}$$

This assumption simplifies the calculation of the likelihood, since we can estimate the probability of each feature independently given the class label, rather than having to estimate the joint probability of all the features.
Since the denominator $P(x_1, x_2, \ldots, x_n)$ is constant across all possible classes, we can ignore it when comparing the relative probabilities of different classes. Therefore, we can calculate the posterior probability as:

$$P(y|x_1, x_2, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y) \tag{3}$$

To calculate the likelihood $P(x_i|y)$ for each feature i we can use the appropriate probability distribution (e.g., a normal distribution for continuous features or the frequency of occurrence for categorical features) or to learn it non parametrically (following TP) .

Likelihood
For continuous features, we can assume that the data follows a normal distribution, and estimate the likelihood $P(x_i|y)$ using the mean and variance of the feature values in class y:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_{i|y}^2}} \exp\left(-\frac{(x_i - \mu_{i|y})^2}{2\sigma_{i|y}^2}\right) \tag{4}$$

where $P(x_i \mid y)$ is the conditional probability of feature x_i given class y, $\mu_{y|i}$ is the mean of feature i for class y, $\sigma_{y|i}$ is the standard deviation of feature i for class y.

For categorical features, we can estimate the probability $P(x_i|y)$ as the frequency of each value v in the training data for a given feature i and class y as:

$$P(x_i = v|y = y_k) = \frac{number\,of\,training\,samples\,with\,feature\,i\,equal\,to\,v\,and\,label\,y_k}{number\,of\,training\,samples\,with\,label\,y_k} \tag{5}$$

Prior
For the prior probability $P(y)$ of class y_k, we simply calculate the proportion of the training samples that belong to class y_k:

$$P(y_k) = \frac{number\,of\,training\,samples\,with\,label\,y_k}{total\,number\,of\,training\,samples} \tag{6}$$

Prediction

To use Naive Bayes for classification, we compute the posterior probability $P(y|x_1, x_2, \ldots, x_n)$ for each possible class label, and choose the class label with the highest probability as the predicted label for the input instance.

Finally, we can predict the class with the highest posterior probability:

$$\hat{y} = \arg\max_y \log P(y|x_1, x_2, \ldots, x_n) \tag{7}$$

Exercises

Your task is to implement the Naive Bayes classifier (with the help of ChatGPT) using only numpy in Python 3, and it should work for continuous datasets (assuming that they follow Normal distribution), categorical datasets, and datasets with both continuous and categorical attributes.

The NaiveBayes() Class should contain the following functions:

```
def __init__(self):
    """Initializes the NaiveBayes object."""
    pass

def fit(self, X, y):
    """Trains the NaiveBayes object on the input data X and labels y."""
    pass

def predict(self, X):
    """Predicts the labels for the input data X using the trained
    NaiveBayes object."""
    pass

def prior(self, y):
    """Computes the prior probability of each class given the labels y."""
    pass

def normal_distribution(self, x, mean, std):
    """Computes the probability density function of a normal
     distribution with mean and standard deviation."""
    pass
```

- In the__init__(self) method, you can initialize any variables that the class needs. In the fit method, you should implement the training procedure for the Naive Bayes classifier using the input data X and corresponding labels y.
- In the predict method, you should implement the prediction procedure for the Naive Bayes classifier, which takes an input data matrix X and returns a vector of predicted labels.
- The prior method takes a vector of labels y and returns a vector of prior probabilities for each class. You can use this method to compute the prior probability for each class based on the number of occurrences of each class in the training data.

- The normal_distribution method takes a scalar value x, a mean, and a standard deviation std, and returns the value of the probability density function for a normal distribution with the given mean and standard deviation evaluated at x. You can use this method to compute the likelihood of each feature value given the class.

Once your implementation is ready you will train your algorithm using the iris dataset (load from scikit-learn) and the Titanic dataset (titanic.csv file) and the Adult dataset (adult.csv file).

1. Load the dataset.
2. Clean your data if needed (check for missing data etc).
3. Split the data into training and test sets (80-20), use seed number for reproducibility.
4. Run your algorithm and compute test classification accuracy. Comment your results.
5. Compare the results of the three datasets. Discuss
6. For the iris dataset plot the decision surfaces with the help of ChatGTP. For the visualization purposes choose two attributes as predictive attributes and color the plane defined by these two attributes on the basis of class labels that Naive Bayes predicts, compute the percentage of misclassified instances on the test set. Comment discuss the result.
   (a) Load a iris dataset and keep only two continuous attributes.
   (b) Split the new iris dataset into training and testing sets.
   (c) Train the Naive Bayes classifier on the training set.
   (d) Generate a grid of points in the feature space.
   (e) Compute the predicted labels for each point in the grid using the Naive Bayes classifier.
   (f) Plot the decision surfaces and the training data using matplotlib library

Datasets

Iris dataset
Iris has 4 numerical features and a tri class target variable. It consists 4 features (attributes) sepal length, sepal width, petal length and petal width and the target variable has 3 classes namely 'setosa', 'versicolor', and 'virginica'. See here for for more information on this dataset.

Adult income dataset An individual's annual income results from various factors. Intuitively, it is influenced by the individual's education level, age, gender, occupation, and etc. The dataset contains 16 columns.

- Target filed: Income
  - The income is divide into two classes: $\leq$50K and >50K
- Number of attributes: 14 (continuous and categorical)
  - These are the demographics and other features to describe a person
- The detailed description on the dataset can be found in the original UCI documentation here.